FIG. 1

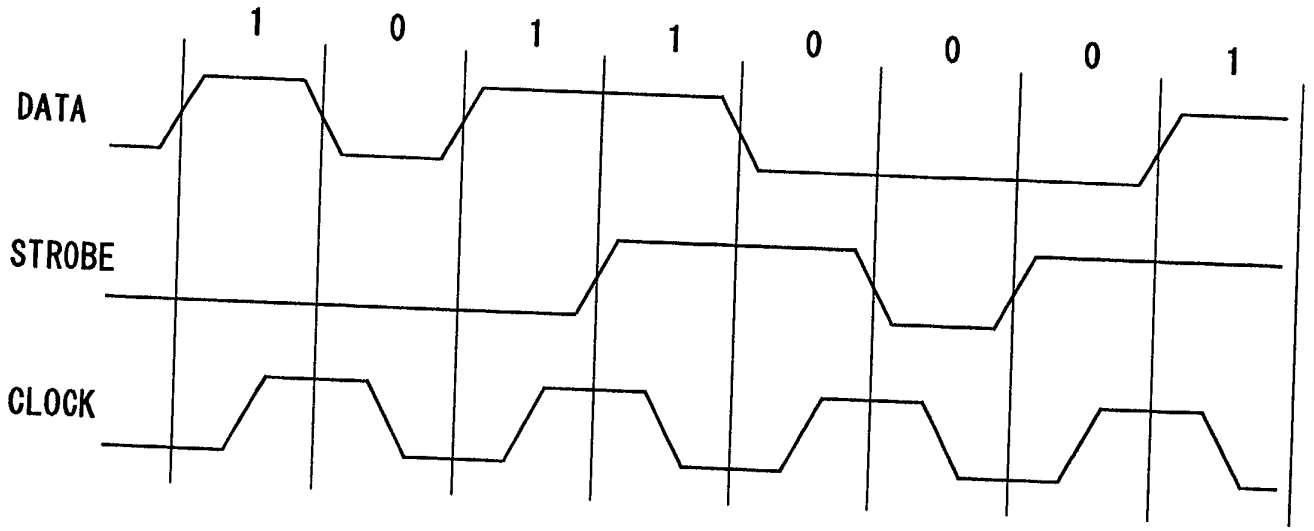


FIG. 2

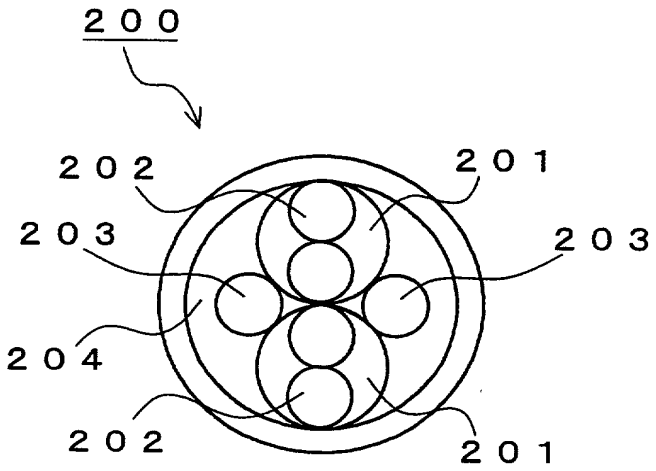


FIG. 3A
(BUS INITIALIZATION)

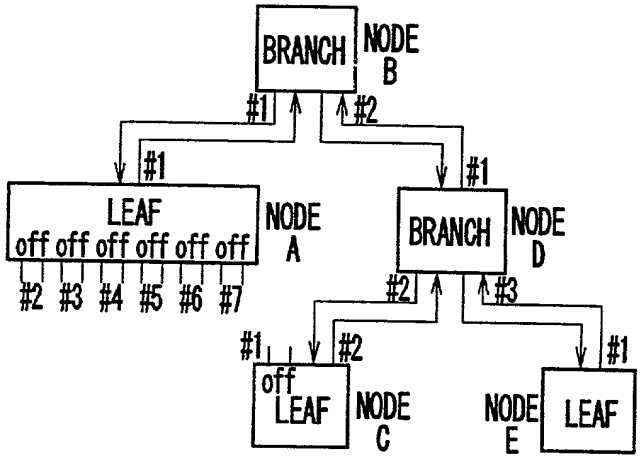


FIG. 3B
(TREE IDENTIFICATION)

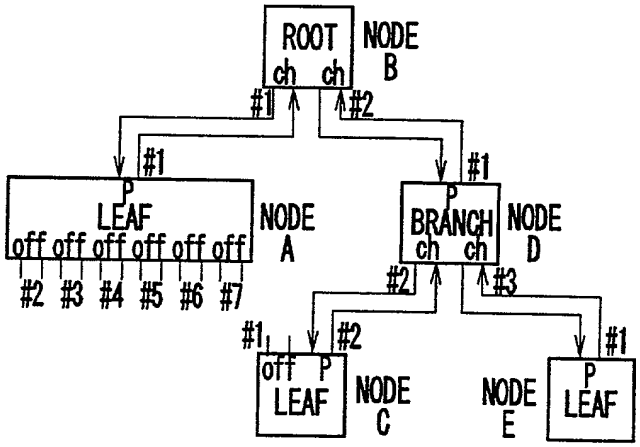
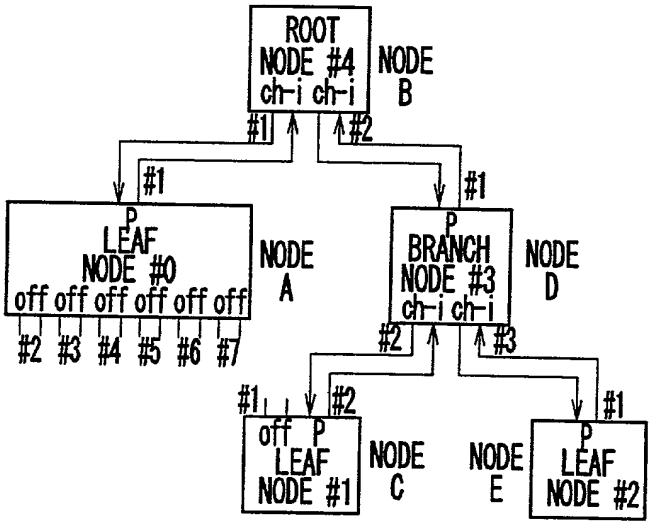


FIG. 3C
(SELF IDENTIFICATION)



U.S. Patent Oct. 26, 1999 Sheet 3 of 5 5,974,501

FIG. 4

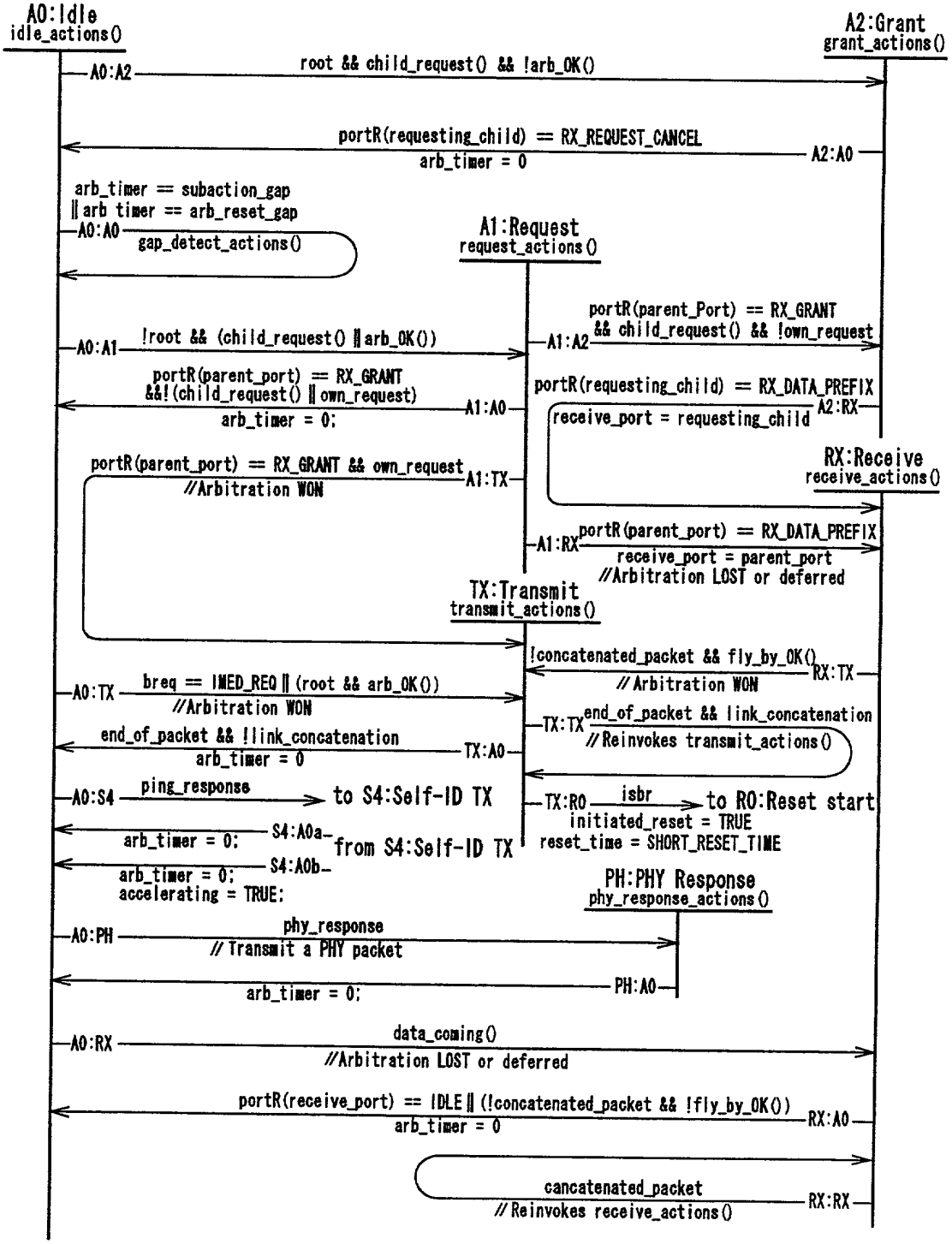

A0:Idle
idle_actions()
A2:Grant
grant_actions()
A0:A2
root && child_request() && !arb_OK()
portR(requesting_child) == RX_REQUEST_CANCEL
arb_timer = 0
A2:A0
arb_timer == subaction_gap
|| arb_timer == arb_reset_gap
A0:A0
gap_detect_actions()
A1:Request
request_actions()
portR(parent_Port) == RX_GRANT
&& child_request() && !own_request
A0:A1
!root && (child_request() || arb_OK())
A1:A2
portR(parent_port) == RX_GRANT
&&!(child_request() || own_request)
arb_timer = 0;
A1:A0
portR(requesting_child) == RX_DATA_PREFIX
receive_port = requesting_child
A2:RX
RX:Receive
receive_actions()
portR(parent_port) == RX_GRANT && own_request
//Arbitration WON
A1:TX
A1:RX
portR(parent_port) == RX_DATA_PREFIX
receive_port = parent_port
//Arbitration LOST or deferred
TX:Transmit
transmit_actions()
!concatenated_packet && fly_by_OK()
//Arbitration WON
RX:TX
A0:TX
breq == IMED_REQ || (root && arb_OK())
//Arbitration WON
TX:TX
end_of_packet && !link_concatenation
//Reinvokes transmit_actions()
end_of_packet && !link_concatenation
arb_timer = 0
TX:A0
A0:S4
ping_response
to S4:Self-ID TX
TX:R0
isbr
to R0:Reset start
initiated_reset = TRUE
reset_time = SHORT_RESET_TIME
arb_timer = 0;
S4:A0a
from S4:Self-ID TX
arb_timer = 0;
accelerating = TRUE;
S4:A0b
PH:PHY Response
phy_response_actions()
A0:PH
phy_response
//Transmit a PHY packet
arb_timer = 0;
PH:A0
A0:RX
data_coming()
//Arbitration LOST or deferred
portR(receive_port) == IDLE || (!concatenated_packet && !fly_by_OK())
arb_timer = 0
RX:A0
cancatenated_packet
//Reinvokes receive_actions()
RX:RX

FIG. 5

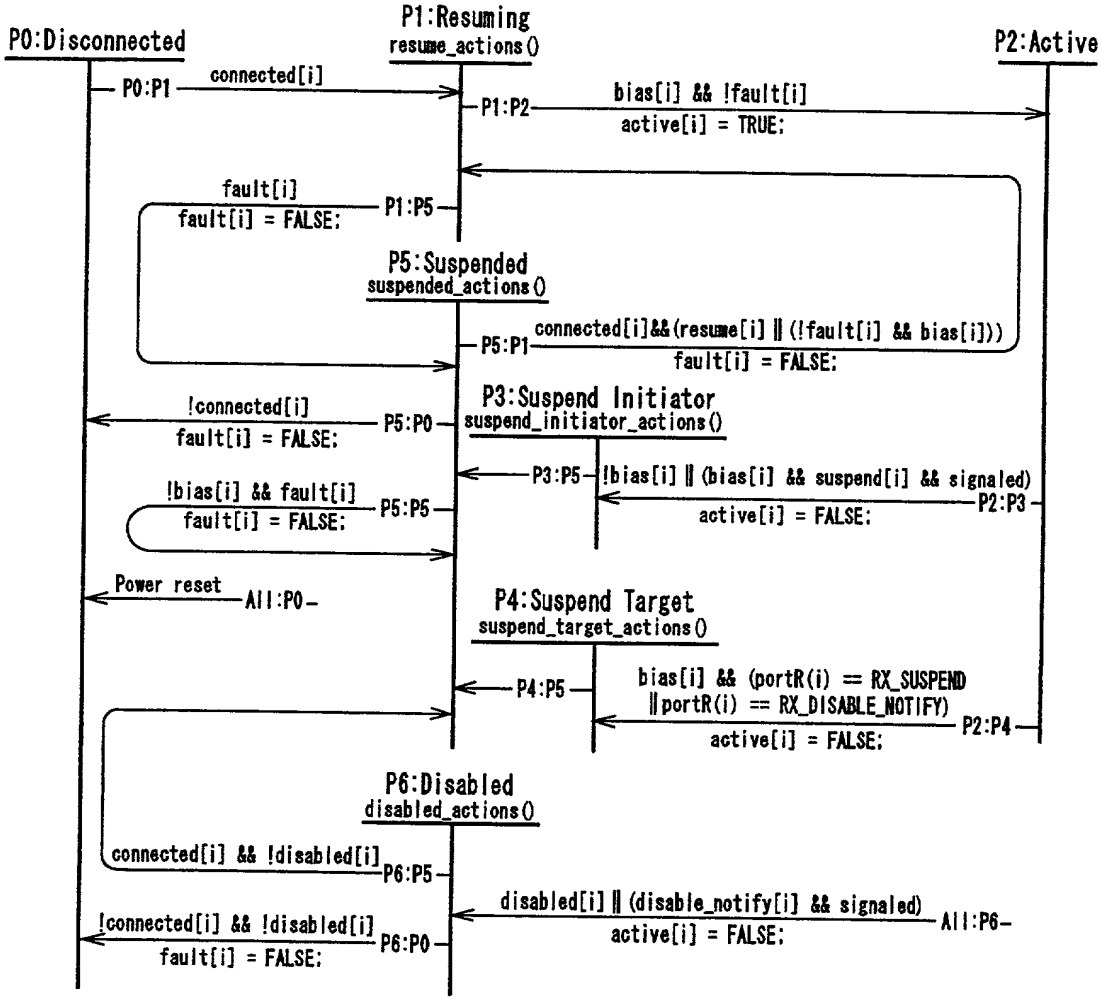

P0:Disconnected
P1:Resuming
resume_actions()
P2:Active
P0:P1
connected[i]
P1:P2
bias[i] && !fault[i]
active[i] = TRUE;
fault[i]
fault[i] = FALSE;
P1:P5
P5:Suspended
suspended_actions()
P5:P1
connected[i]&&(resume[i] || (!fault[i] && bias[i]))
fault[i] = FALSE;
P3:Suspend Initiator
suspend_initiator_actions()
!connected[i]
fault[i] = FALSE;
P5:P0
P3:P5
!bias[i] || (bias[i] && suspend[i] && signaled)
active[i] = FALSE;
P2:P3
!bias[i] && fault[i]
fault[i] = FALSE;
P5:P5
Power reset
All:P0
P4:Suspend Target
suspend_target_actions()
P4:P5
bias[i] && (portR(i) == RX_SUSPEND
|| portR(i) == RX_DISABLE_NOTIFY)
active[i] = FALSE;
P2:P4
P6:Disabled
disabled_actions()
connected[i] && !disabled[i]
P6:P5
disabled[i] || (disable_notify[i] && signaled)
active[i] = FALSE;
All:P6
!connected[i] && !disabled[i]
fault[i] = FALSE;
P6:P0

FIG. 6A

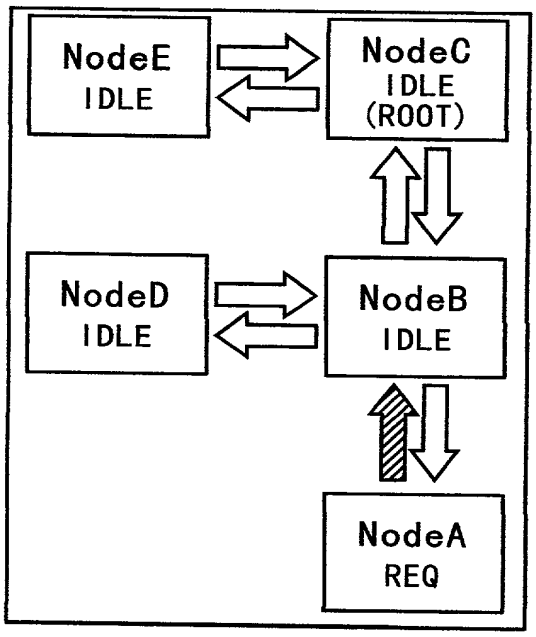


FIG. 6B

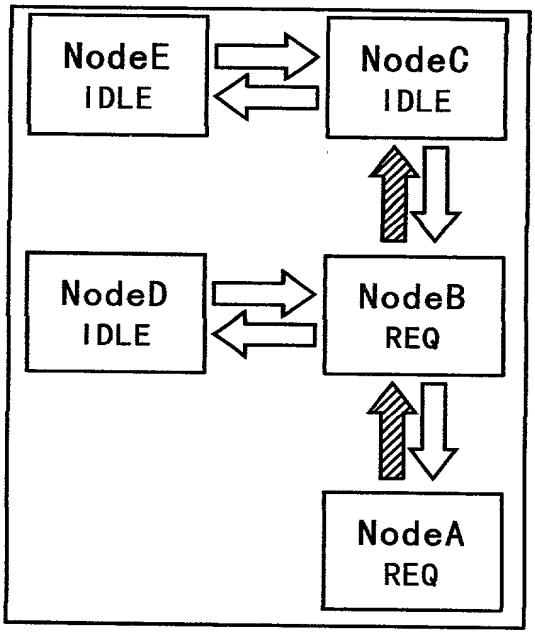


FIG. 6C

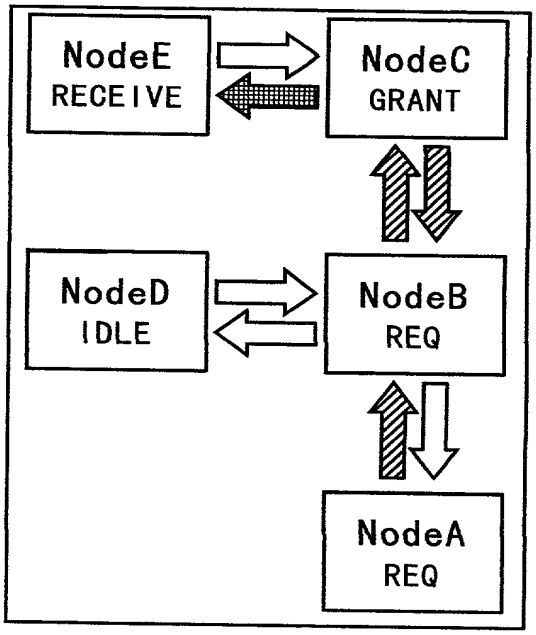


FIG. 6D

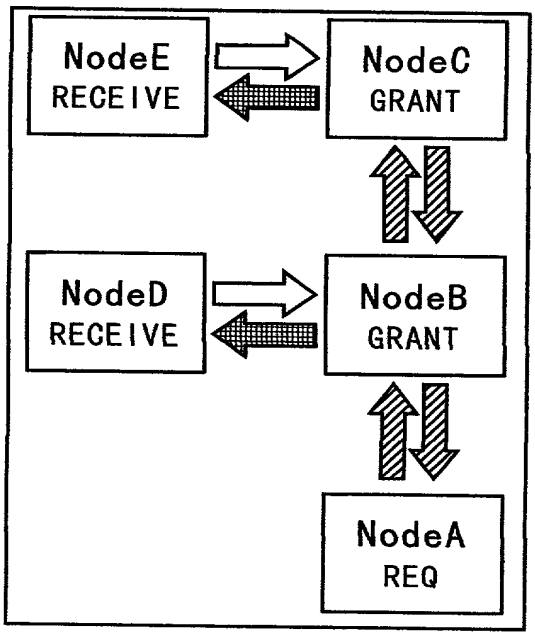


FIG. 7A

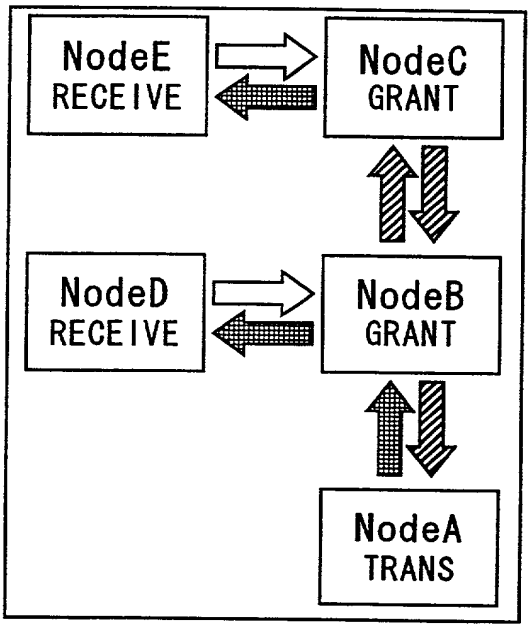


FIG. 7B

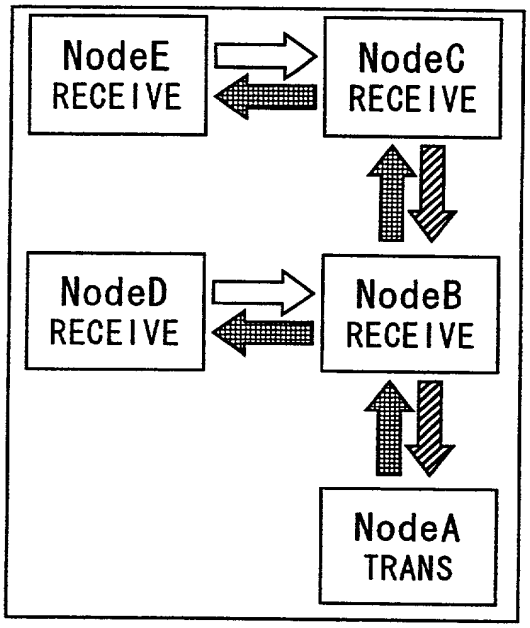


FIG. 7C

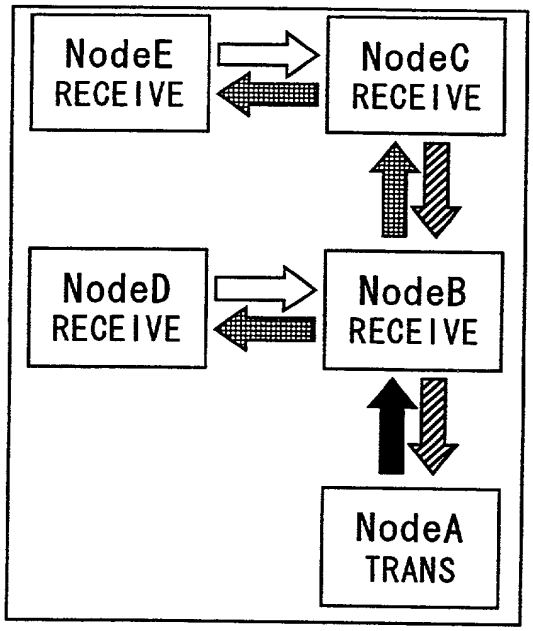


FIG. 7D

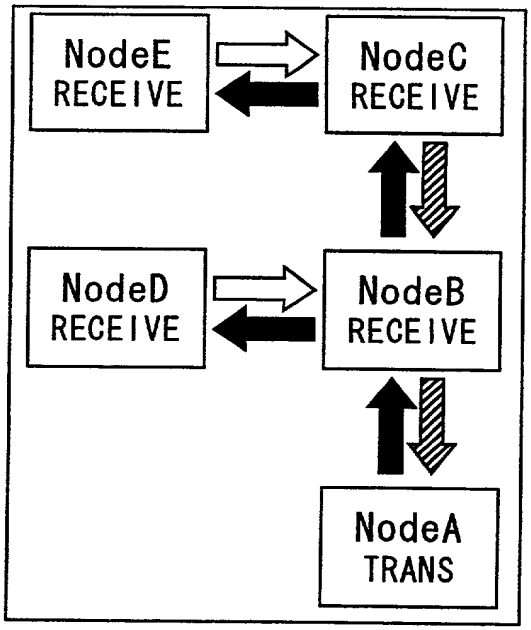


FIG. 8A

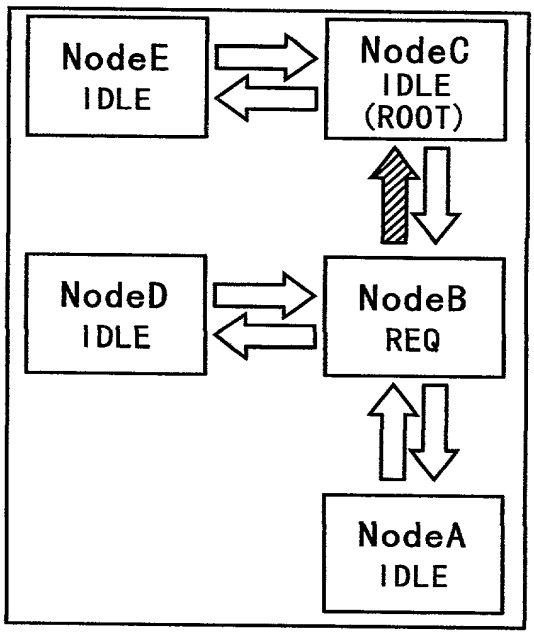


FIG. 8B

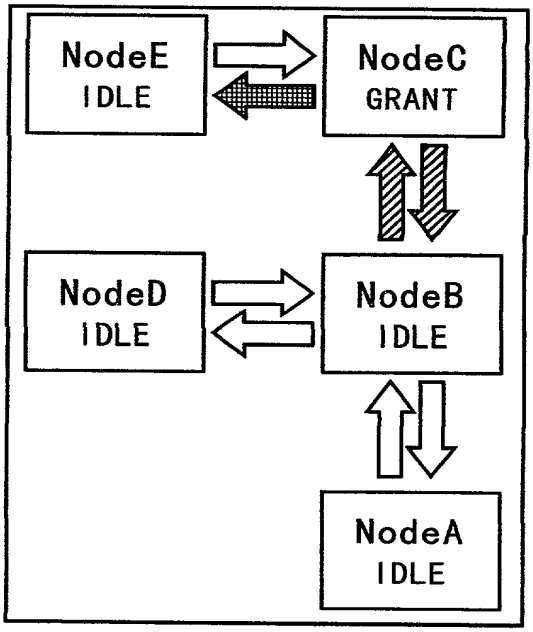


FIG. 8C

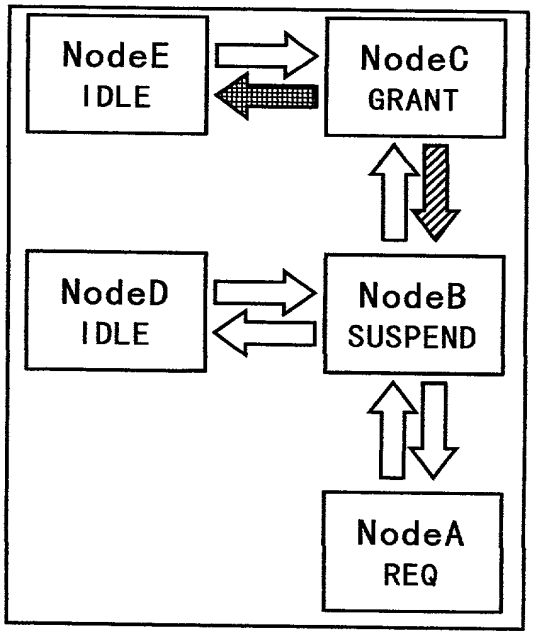


FIG. 9A

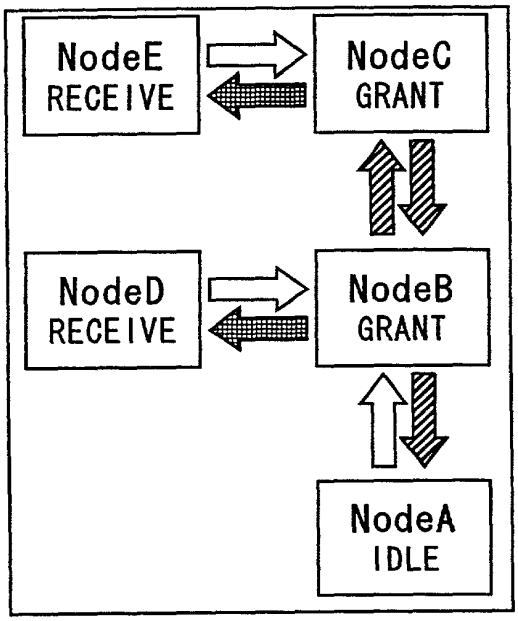


FIG. 9B

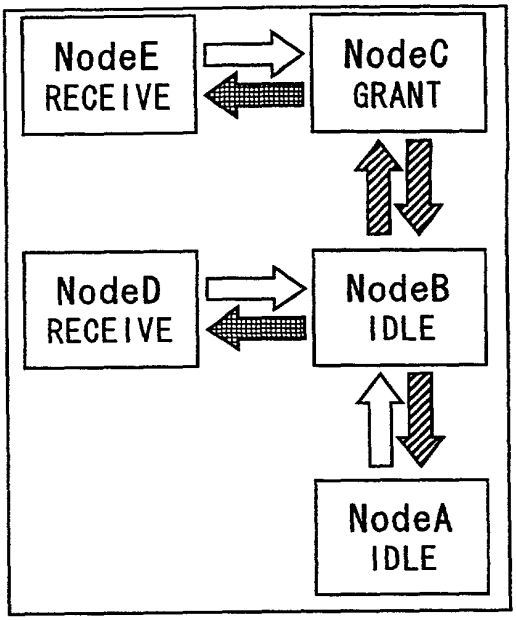


FIG. 9C

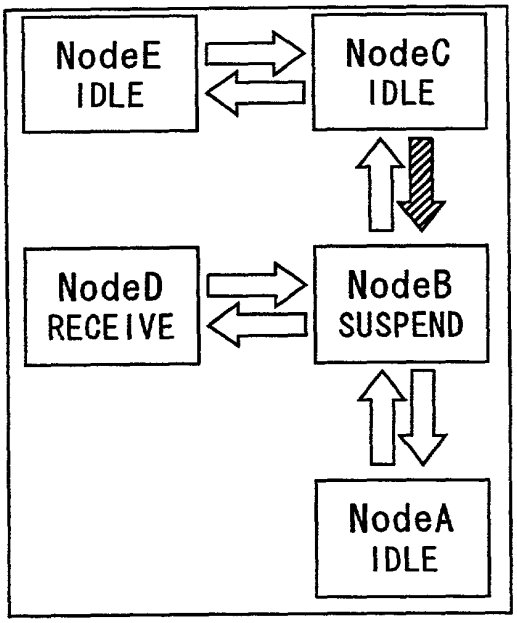


FIG. 10

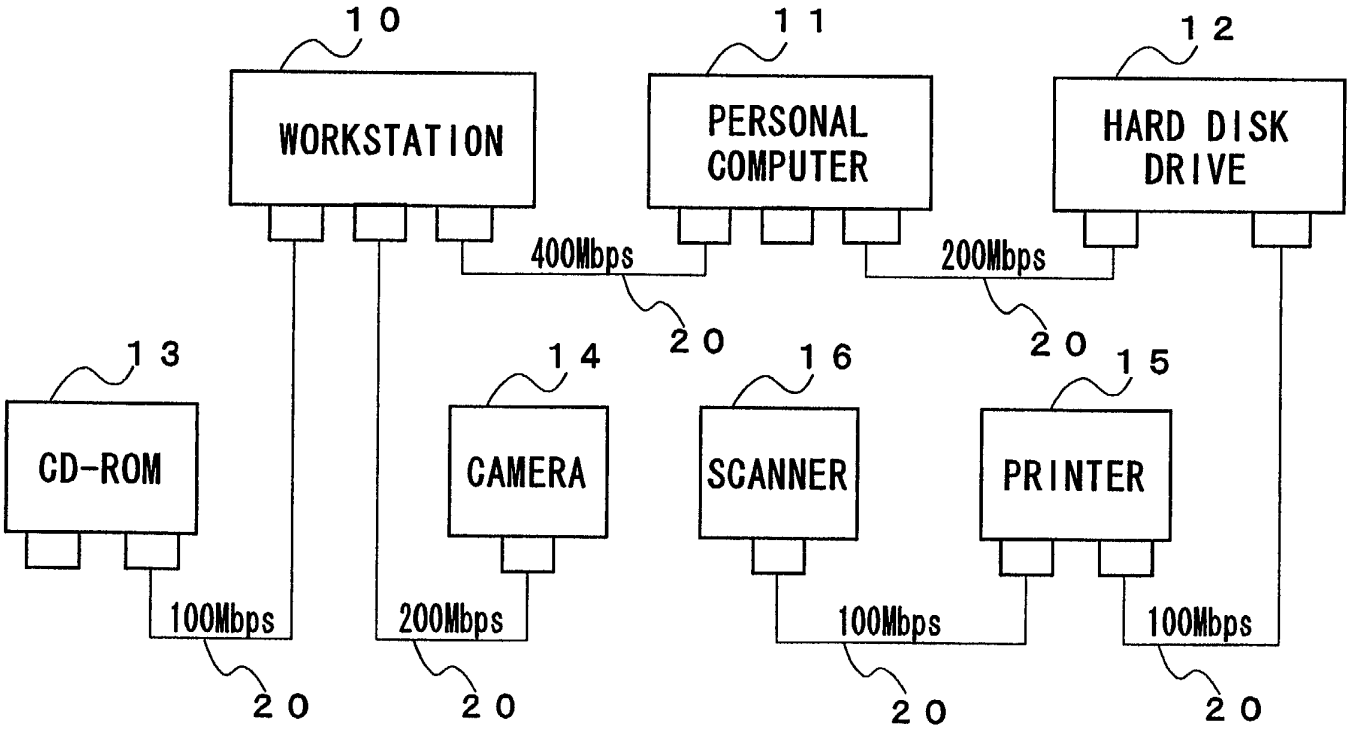

10
WORKSTATION
11
PERSONAL COMPUTER
12
HARD DISK DRIVE
400Mbps
200Mbps
20
20
13
CD-ROM
14
CAMERA
16
SCANNER
15
PRINTER
100Mbps
200Mbps
100Mbps
100Mbps
20
20
20
20

FIG. 11

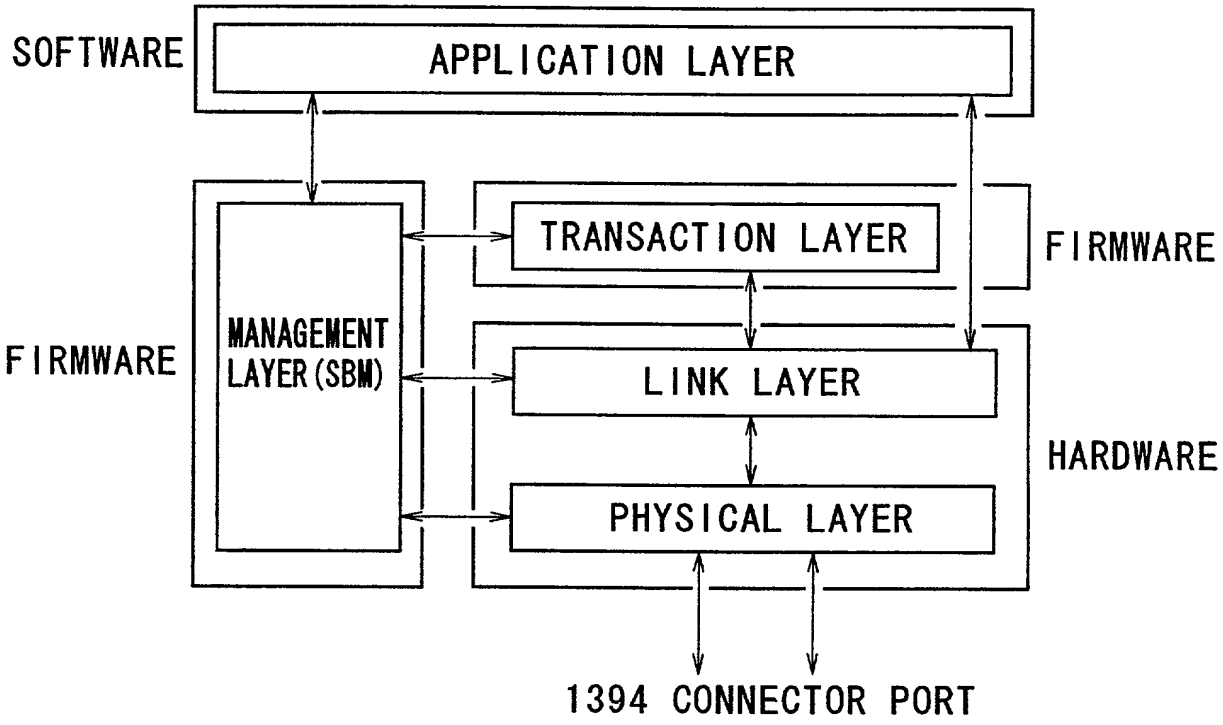

SOFTWARE
APPLICATION LAYER
TRANSACTION LAYER
FIRMWARE
FIRMWARE
MANAGEMENT LAYER(SBM)
LINK LAYER
HARDWARE
PHYSICAL LAYER
1394 CONNECTOR PORT

FIG. 12

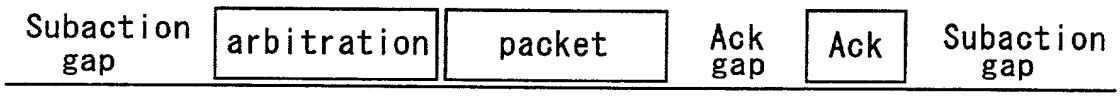

Subaction gap
arbitration
packet
Ack gap
Ack
Subaction gap

FIG. 13A

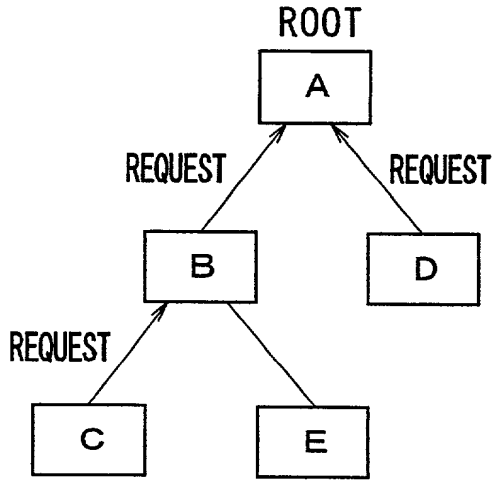


FIG. 13B

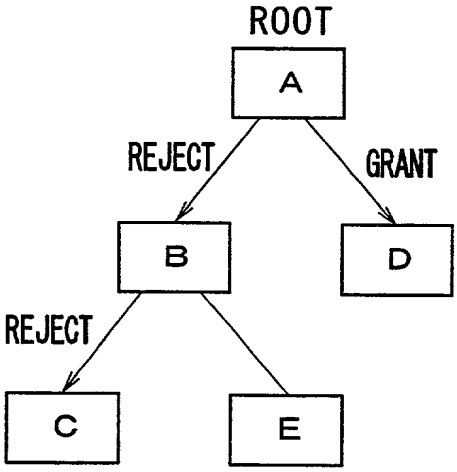


FIG. 14

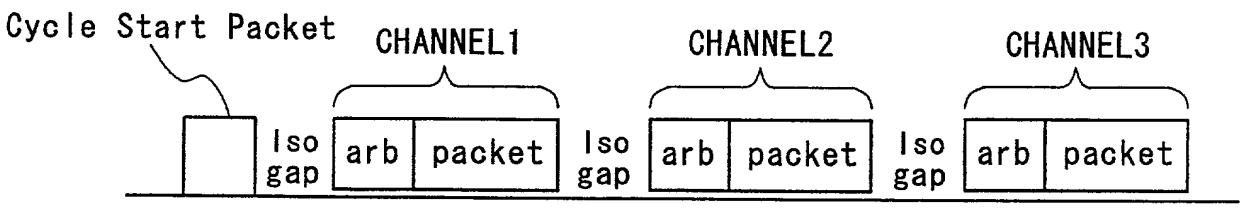


FIG. 15

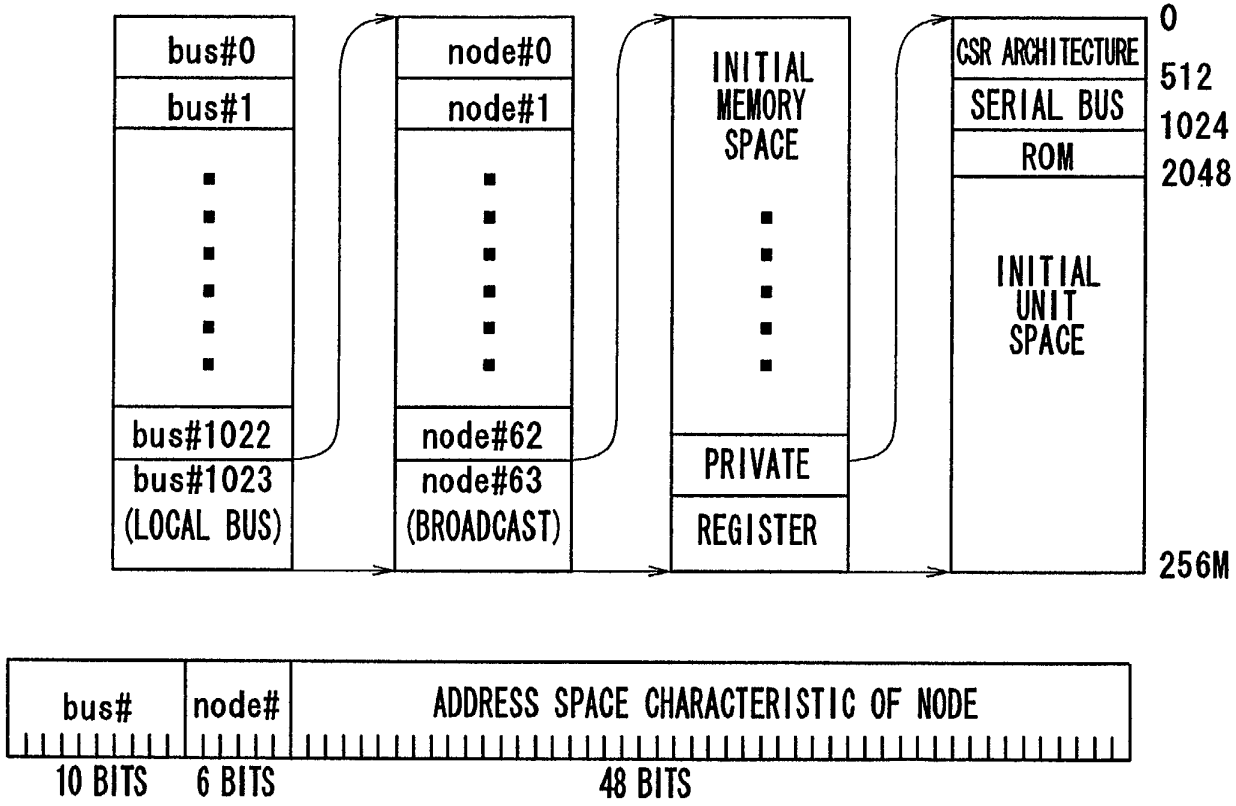

bus#0
bus#1
bus#1022
bus#1023
(LOCAL BUS)
node#0
node#1
node#62
node#63
(BROADCAST)
INITIAL
MEMORY
SPACE
PRIVATE
REGISTER
CSR ARCHITECTURE
SERIAL BUS
ROM
INITIAL
UNIT
SPACE
0
512
1024
2048
256M
bus#
node#
ADDRESS SPACE CHARACTERISTIC OF NODE
10 BITS
6 BITS
48 BITS

FIG. 16

OFFSETS	NAMES	FUNCTIONS
000h	STATE_CLEAR	STATE AND CONTROL INFORMATION
004h	STATE_SET	SET STATE_CLEAR BIT
008h	NODE_IDs	INDICATE 16-BIT NODE ID
00Ch	RESET_START	START COMMAND RESET
018h-01Ch	SPLIT_TIMEOUT	PRESCRIBE MAXIMUM TIME OF SPLIT
200h	CYCLE_TIME	CYCLE TIME
210h	BUSY_TIMEOUT	PRESCRIBE LIMIT OF RETRY
21Ch	BUS_MANAGER	INDICATE BUS MANAGER ID
220h	BANDWIDTH_AVAILABLE	INDICATE BANDWIDTH THAT CAN BE ASSIGNED TO ISOCHRONOUS COMMUNICATION
224h-228h	CHANNELS_AVAILABLE	INDICATE USED STATE OF EACH CHANNEL

FIG. 17

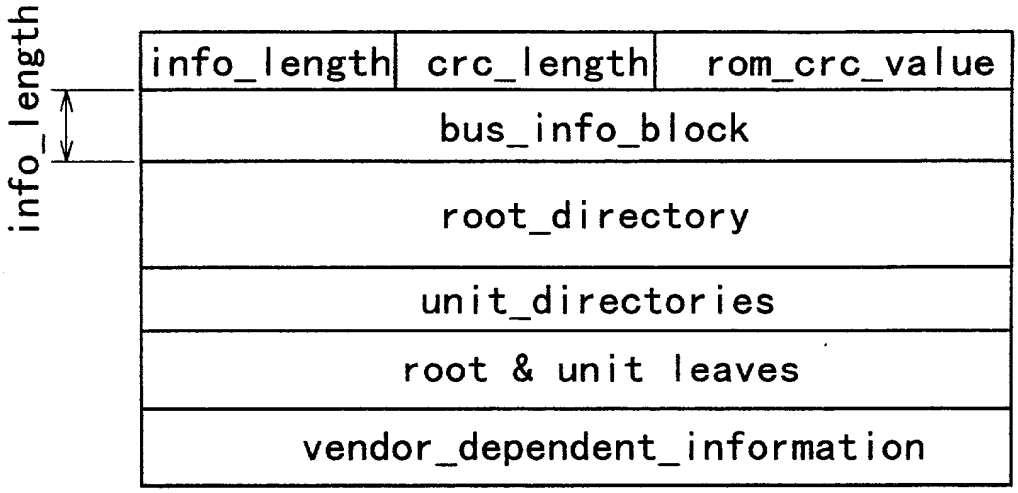


FIG. 19

900h	Output Master Plug Register
904h	Output Plug Control Register #0
908h	Output Plug Control Register #1
⋮	⋮
97Ch	Output Plug Control Register #30
980h	Input Master Plug Register
984h	Input Plug Control Register #0
988h	Input Plug Control Register #1
⋮	⋮
9FCh	Input Plug Control Register #30

FIG. 18

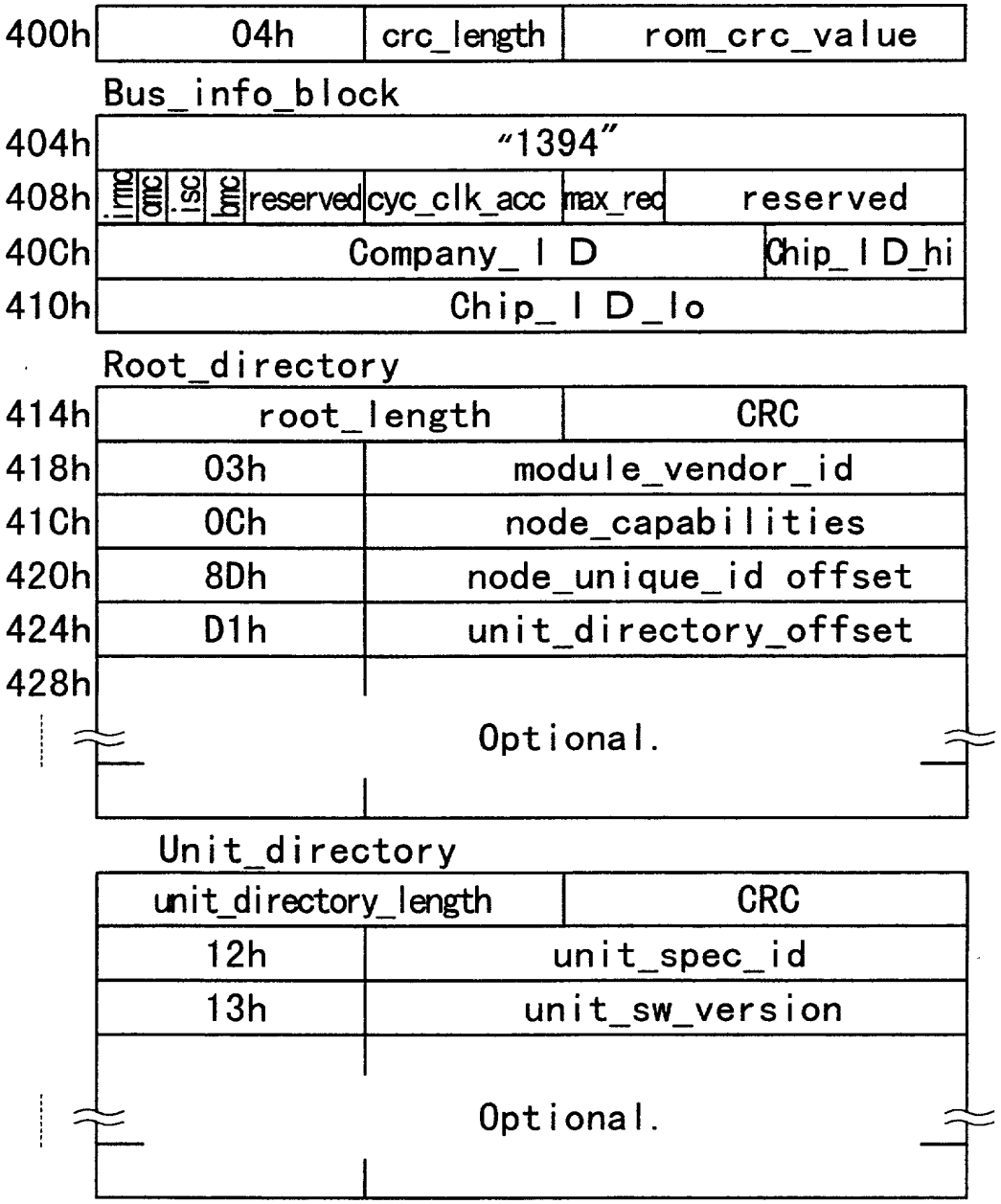

400h
04h
crc_length
rom_crc_value
Bus_info_block
404h
"1394"
408h
irmc
cmc
isc
bmc
reserved
cyc_clk_acc
max_rec
reserved
40Ch
Company_ID
Chip_ID_hi
410h
Chip_ID_lo
Root_directory
414h
root_length
CRC
418h
03h
module_vendor_id
41Ch
0Ch
node_capabilities
420h
8Dh
node_unique_id offset
424h
D1h
unit_directory_offset
428h
Optional.
Unit_directory
unit_directory_length
CRC
12h
unit_spec_id
13h
unit_sw_version
Optional.

FIG. 20A

oMPR

data rate capability	broadcast channel base	non-persistent extension field	persistent extension field	reserved	number of output plugs
2	6	8	8	3	5 (bit)

FIG. 20B

oPCR[n]

on-line	broadcast connection counter	point-to-point connection counter	reserved	channel number	data rate	overhead ID	payload
1	1	6	2	6	2	4	10(bit)

FIG. 20C

iMPR

data rate capability	reserved	non-persistent extension field	persistent extension field	reserved	number of output plugs
2	6	8	8	3	5 (bit)

FIG. 20D

iPCR[n]

on-line	broadcast connection counter	point-to-point connection counter	reserved	channel number	reserved
1	1	6	2	6	16 (bit)

FIG. 21

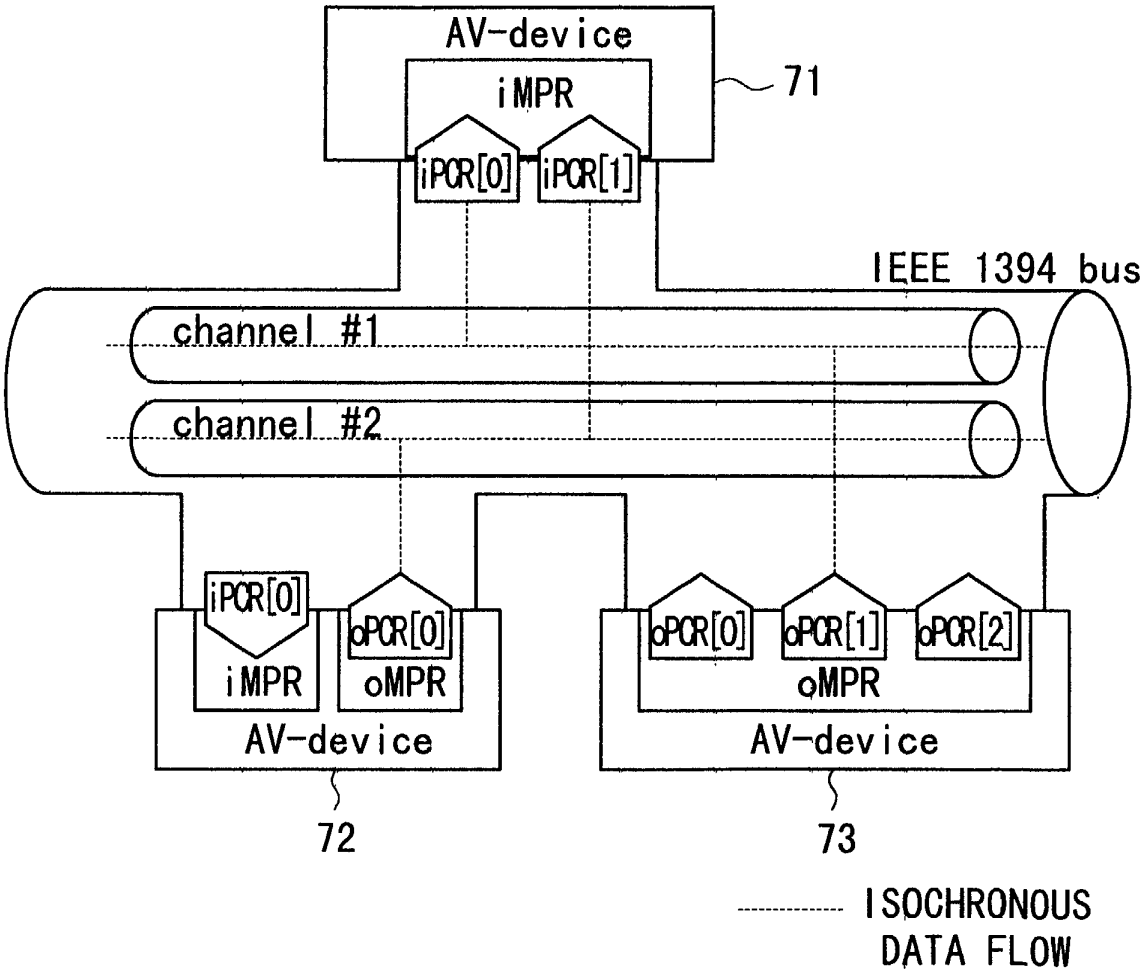

AV-device
iMPR
71
iPCR[0]
iPCR[1]
IEEE 1394 bus
channel #1
channel #2
iPCR[0]
iMPR
oPCR[0]
oMPR
AV-device
72
oPCR[0]
oPCR[1]
oPCR[2]
oMPR
AV-device
73
ISOCHRONOUS DATA FLOW

FIG. 22

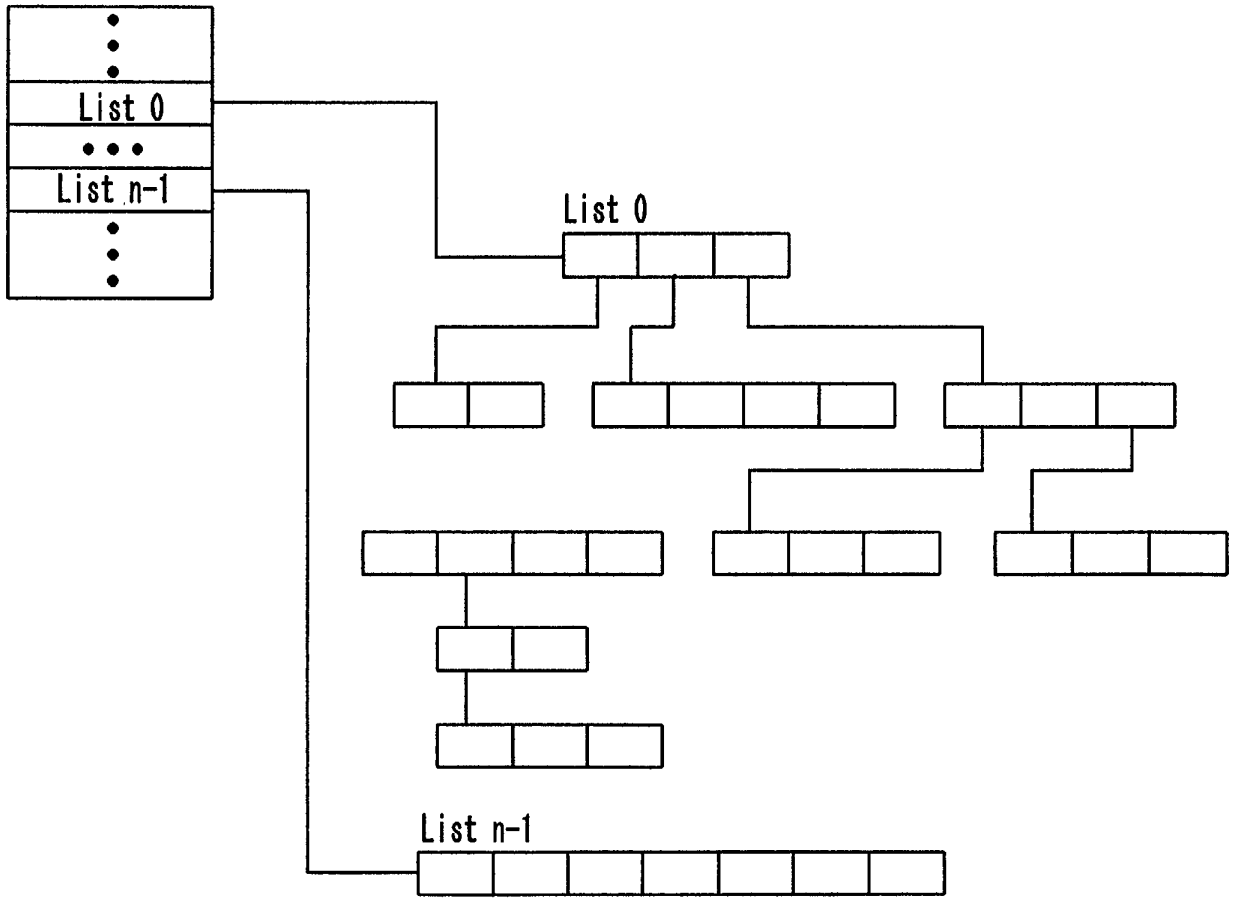


FIG. 23

The General Subunit Identifier Descriptor	
address	contents
00 00_{16}	descriptor_length
00 01_{16}	
00 02_{16}	generation_ID
00 03_{16}	size_of_list_ID
00 04_{16}	size_of_object_ID
00 05_{16}	size_of_object_position
00 06_{16}	number_of_root_object_lists(n)
00 07_{16}	
00 08_{16}	root_object_list_id_0
⋮	
⋮	⋮
⋮	root_object_list_id_n-1
⋮	
⋮	subunit_dependent_length
⋮	
⋮	subunit_dependent_information
⋮	
⋮	
⋮	manufacturer_dependent_length
⋮	
⋮	manufacturer_dependent_information
⋮	
⋮	

Patent Application Publication Jan. 26, 2006 Sheet 23 of 23 US 2006/0020744 A1

FIG. 24

generation_ID values	
generation_ID	meaning
00_{16}	Data structures and command sets as specified in the AV/C General Specification, version 3.0
all others	reserved for future specification

FIG. 25

List ID Value Assignment Ranges	
range of values	list definition
0000_{16}-$0FFF_{16}$	reserved
1000_{16}-$3FFF_{16}$	subunit-type dependent
4000_{16}-$FFFF_{16}$	reserved
1 0000_{16}-max list ID value	subunit-type dependent

FIG. 26

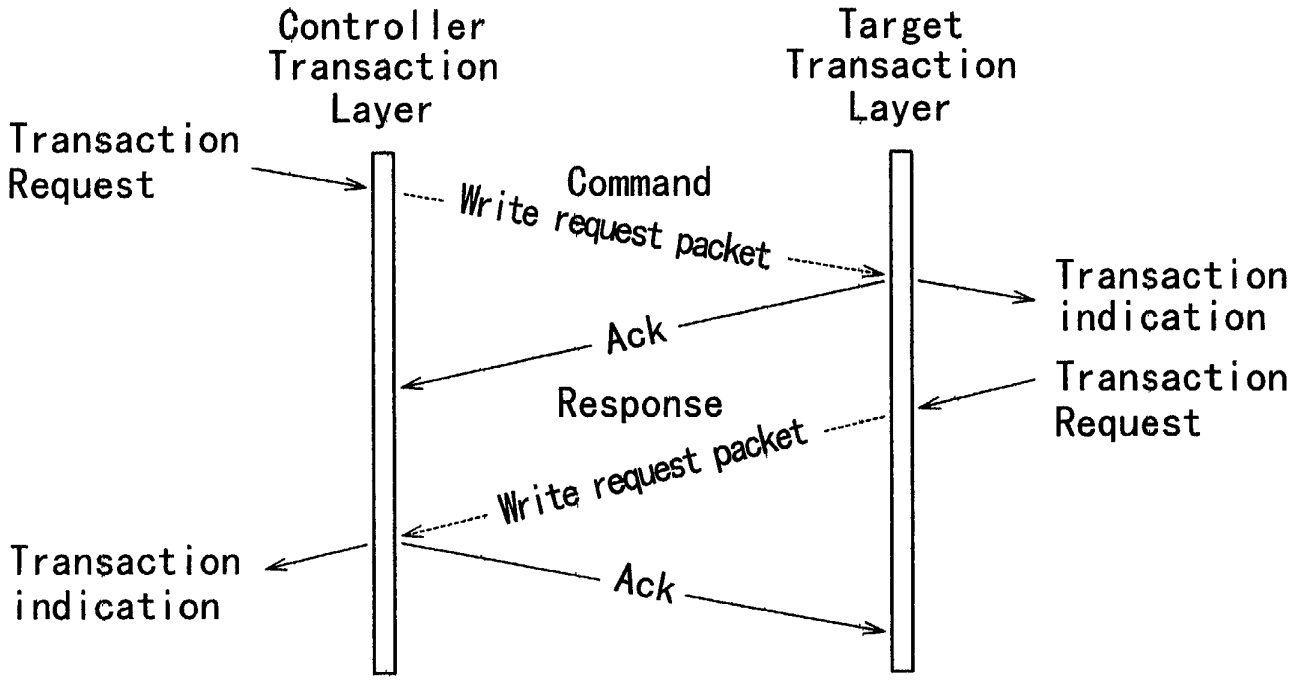

Controller Transaction Layer
Target Transaction Layer
Transaction Request
Command
Write request packet
Transaction indication
Ack
Transaction Request
Response
Write request packet
Transaction indication
Ack

FIG. 27

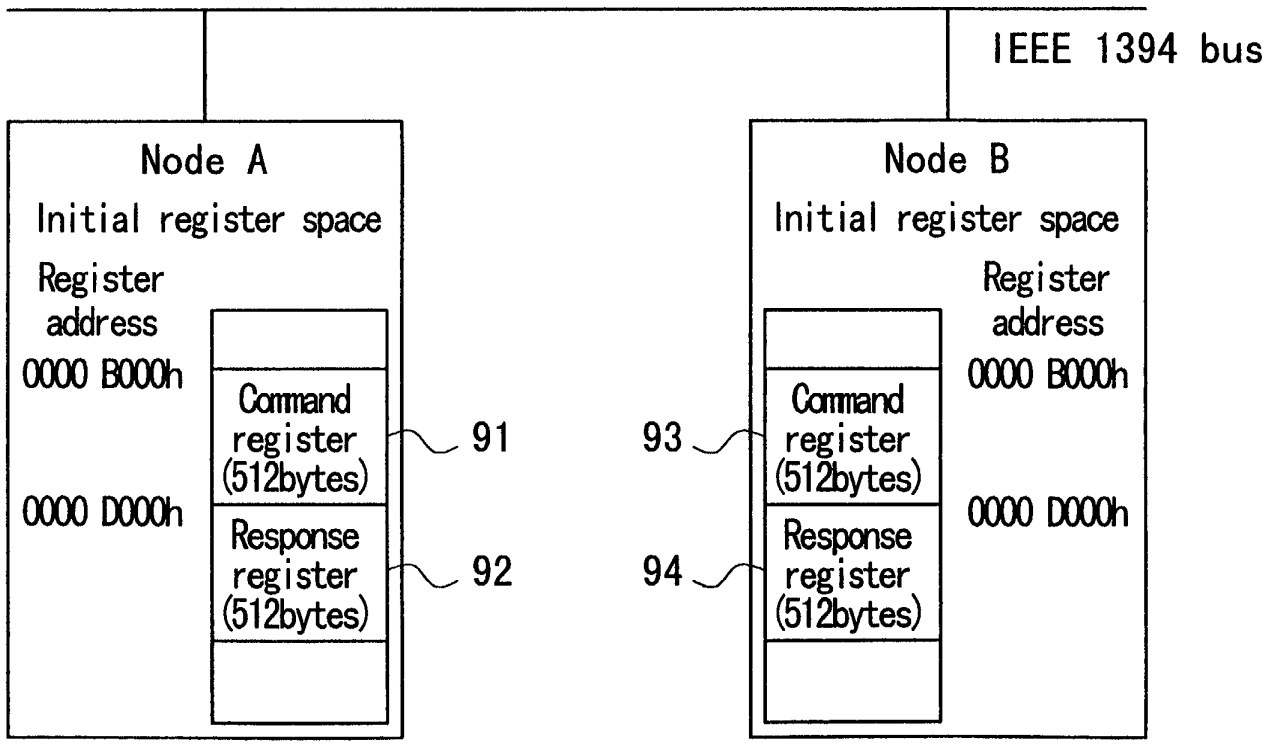
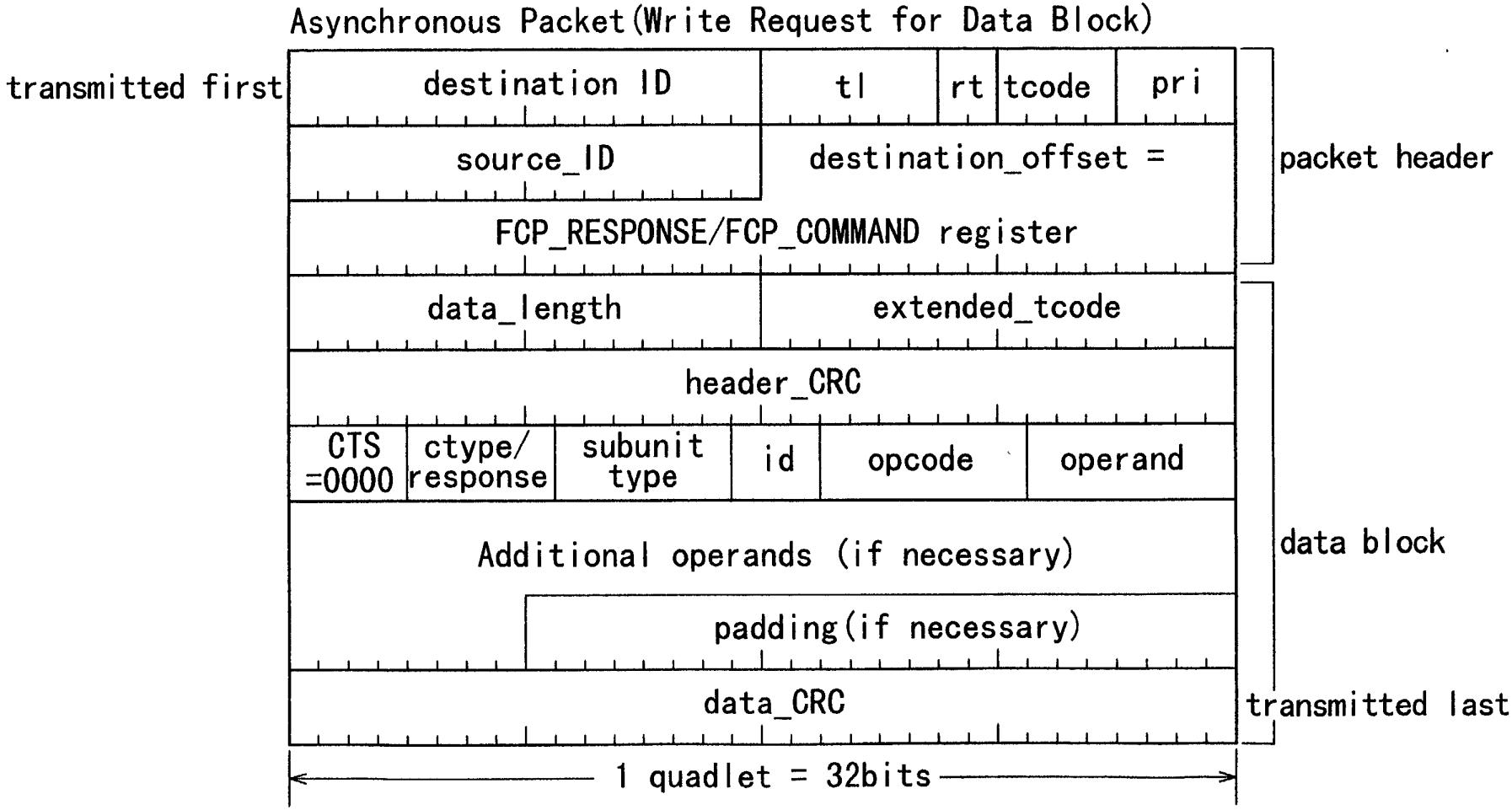

IEEE 1394 bus
Node A
Initial register space
Register address
0000 B000h
0000 D000h
Command register (512bytes)
Response register (512bytes)
91
92
Node B
Initial register space
Register address
0000 B000h
0000 D000h
Command register (512bytes)
Response register (512bytes)
93
94

FIG. 28


Asynchronous Packet(Write Request for Data Block)
transmitted first
destination ID
tl
rt
tcode
pri
source_ID
destination_offset =
FCP_RESPONSE/FCP_COMMAND register
data_length
extended_tcode
header_CRC
CTS =0000
ctype/ response
subunit type
id
opcode
operand
Additional operands (if necessary)
padding(if necessary)
data_CRC
packet header
data block
transmitted last
1 quadlet = 32bits

ctype/response

Command	0000	CONTROL
	0001	STATUS
	0010	SPECIFIC INQUIRY
	0011	NOTIFY
	0100	GENERAL INQUIRY
	0101 ⟨ 0111	(reserved for future specification)
Response	1000	NOT IMPLEMENTED
	1001	ACCEPTED
	1010	REJECTED
	1011	IN TRANSITION
	1100	IMPLEMENTED/STABLE
	1101	CHANGED
	1110	(reserved for future specification)
	1111	INTERIM

FIG. 29A

subunit_type

00000 ⟨ 00011	Video monitor (reserved) Disc recorder/ Player
00100	Tape recorder/ Player
00101	Tuner
00111 ⟨ 11100	Video Camera (reserved) Vendor unique
11101	reserved
11110	Subunit type extended to next byte
11111	Unit

FIG. 29B

opcode:Operation Code

00h	VENDOR-DEPENDENT
50h	SEARCH MODE
51h	TIMECODE
52h	ATN
60h	OPEN MIC
61h	READ MIC
62h	WRITE MIC
C1h	LOAD MEDIUM
C2h	RECORD
C3h	PLAY
C4h	WIND
⟨	⟨

FIG. 29C

FIG. 30A

AV/C	control	tape recorder /player	IN THE CASE OF IDO	PLAY	FORWARD
CTS= 0000	ctype= 0000	subunit type= 00100	id= 000	opcode= C3h	operand= 75h

FIG. 30B

AV/C	accepted	tape recorder /player	IN THE CASE OF IDO	PLAY	FORWARD
CTS= 0000	response =1001	subunit type= 00100	id= 000	opcode= C3h	operand= 75h

FIG. 31

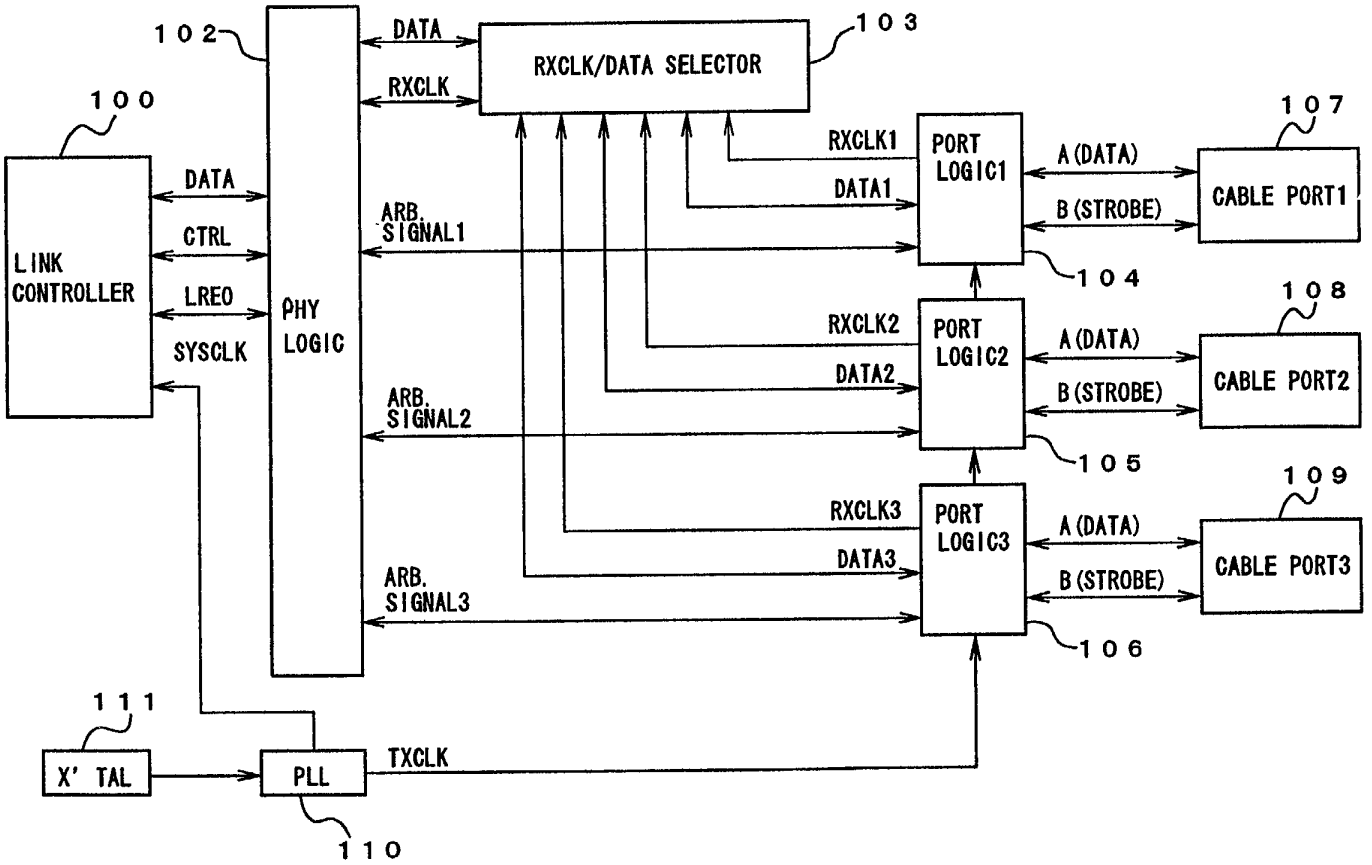

102
DATA
RXCLK
103
RXCLK/DATA SELECTOR
100
LINK CONTROLLER
DATA
CTRL
LREQ
SYSCLK
PHY LOGIC
ARB. SIGNAL1
ARB. SIGNAL2
ARB. SIGNAL3
RXCLK1
DATA1
PORT LOGIC1
A (DATA)
B (STROBE)
104
107
CABLE PORT1
RXCLK2
DATA2
PORT LOGIC2
105
108
CABLE PORT2
RXCLK3
DATA3
PORT LOGIC3
106
109
CABLE PORT3
111
X' TAL
PLL
TXCLK
110

FIG. 32

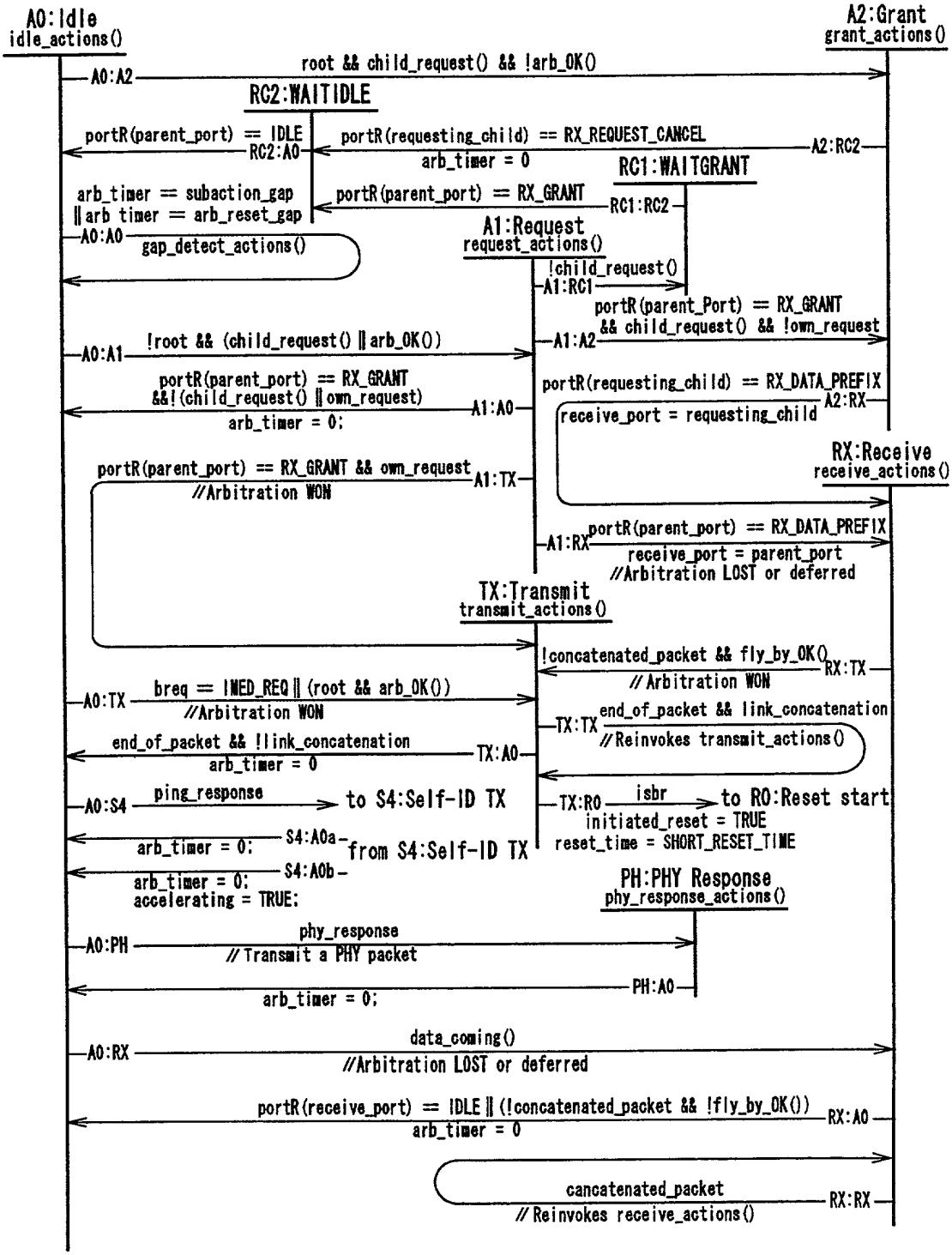

A0:Idle
idle_actions()
A2:Grant
grant_actions()
A0:A2
root && child_request() && !arb_OK()
RC2:WAITIDLE
portR(parent_port) == IDLE
RC2:A0
portR(requesting_child) == RX_REQUEST_CANCEL
arb_timer = 0
A2:RC2
RC1:WAITGRANT
arb_timer == subaction_gap
|| arb_timer == arb_reset_gap
portR(parent_port) == RX_GRANT
RC1:RC2
A0:A0
gap_detect_actions()
A1:Request
request_actions()
!child_request()
A1:RC1
portR(parent_Port) == RX_GRANT
&& child_request() && !own_request
A0:A1
!root && (child_request() || arb_OK())
A1:A2
portR(parent_port) == RX_GRANT
&&!(child_request() || own_request)
arb_timer = 0;
A1:A0
portR(requesting_child) == RX_DATA_PREFIX
A2:RX
receive_port = requesting_child
RX:Receive
receive_actions()
portR(parent_port) == RX_GRANT && own_request
//Arbitration WON
A1:TX
A1:RX
portR(parent_port) == RX_DATA_PREFIX
receive_port = parent_port
//Arbitration LOST or deferred
TX:Transmit
transmit_actions()
!concatenated_packet && fly_by_OK()
//Arbitration WON
RX:TX
A0:TX
breq == IMED_REQ || (root && arb_OK())
//Arbitration WON
TX:TX
end_of_packet && link_concatenation
//Reinvokes transmit_actions()
end_of_packet && !link_concatenation
arb_timer = 0
TX:A0
A0:S4
ping_response
to S4:Self-ID TX
TX:R0
isbr
to R0:Reset start
initiated_reset = TRUE
reset_time = SHORT_RESET_TIME
arb_timer = 0;
S4:A0a
from S4:Self-ID TX
arb_timer = 0;
accelerating = TRUE;
S4:A0b
PH:PHY Response
phy_response_actions()
A0:PH
phy_response
//Transmit a PHY packet
PH:A0
arb_timer = 0;
A0:RX
data_coming()
//Arbitration LOST or deferred
portR(receive_port) == IDLE || (!concatenated_packet && !fly_by_OK())
arb_timer = 0
RX:A0
cancatenated_packet
//Reinvokes receive_actions()
RX:RX

FIG. 33A

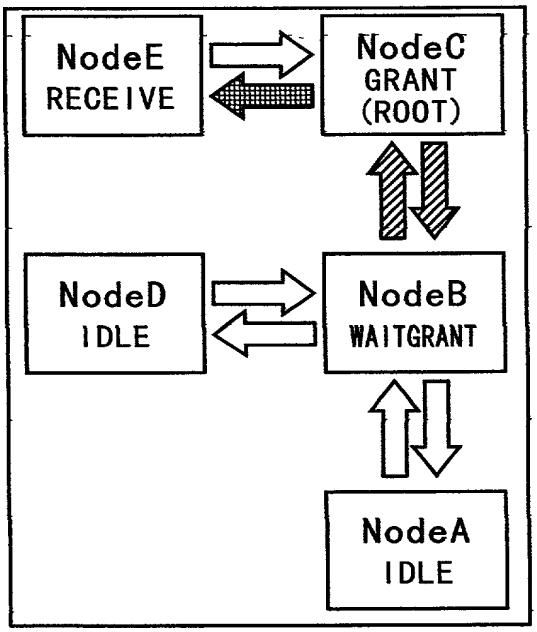


FIG. 33B

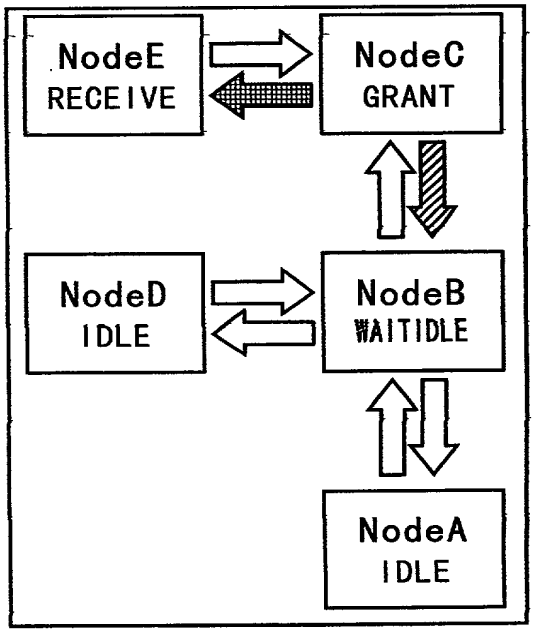


FIG. 33C

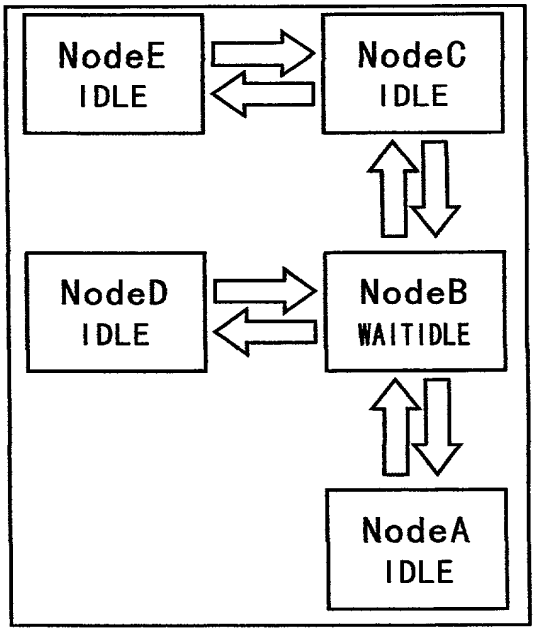


FIG. 33D

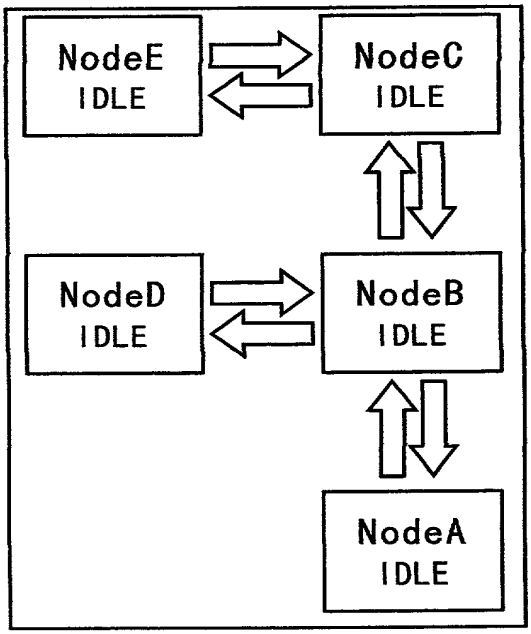


FIG. 34A

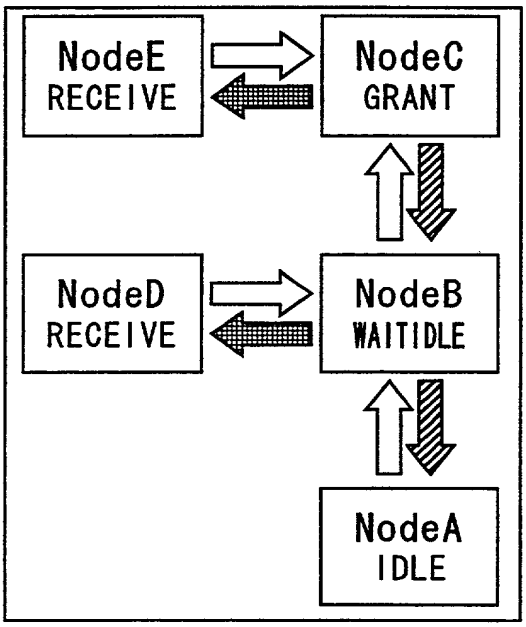


FIG. 34B

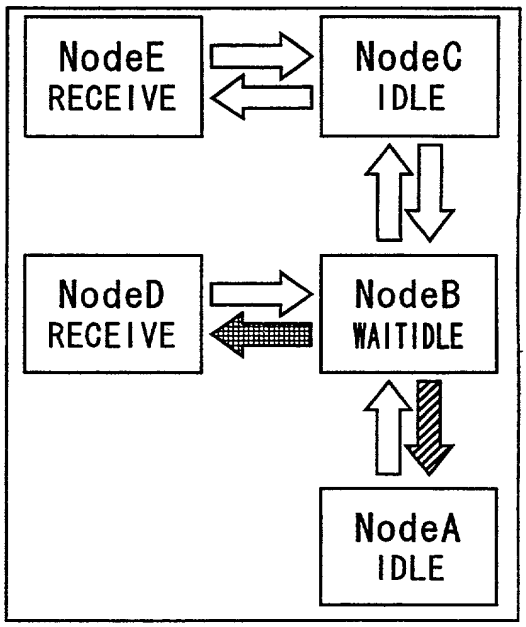
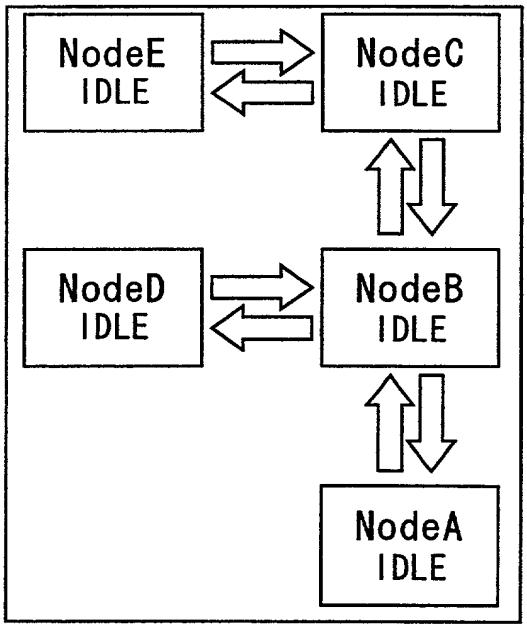



FIG. 34C